CSE 559A: Computer Vision

[credit: danjodon.deviantart.com]

Fall 2017: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Staff: Abby Stylianou (abby@wustl.edu), Jarett Gross (jarett@wustl.edu)

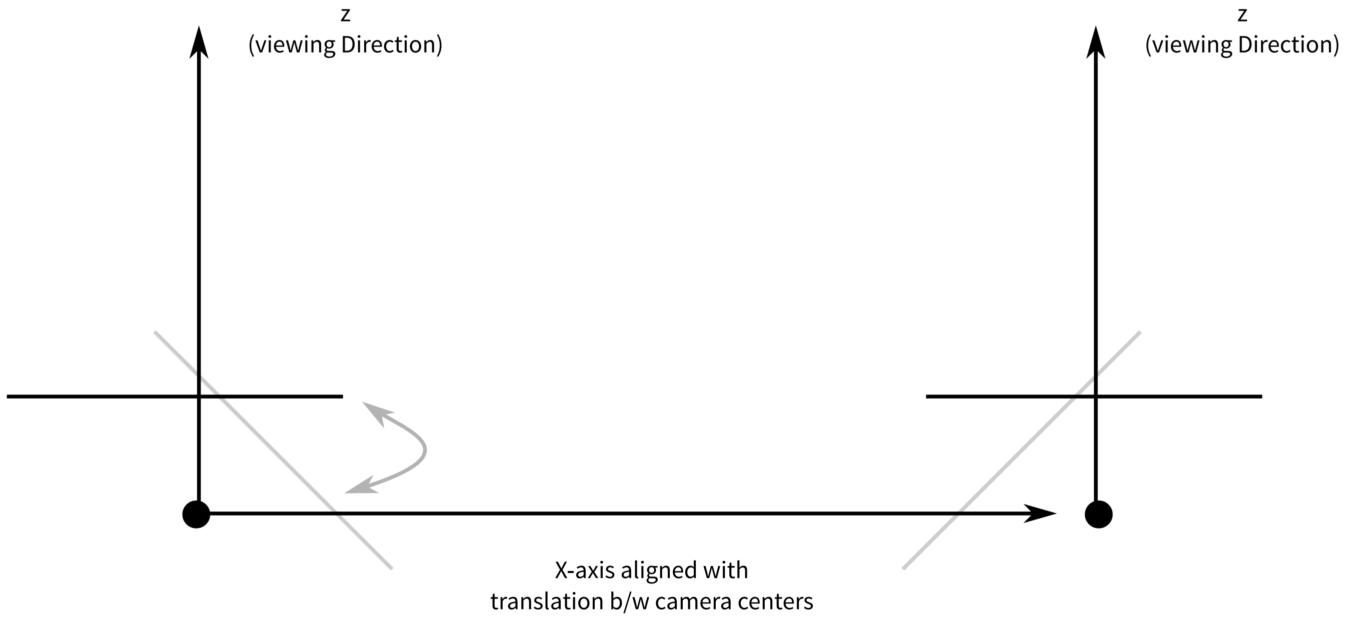
http://www.cse.wustl.edu/~ayan/courses/cse559a/

Oct 19, 2017

GENERAL

- Problem set 3 due next Thursday.
- Recitation Tomorrow.
- Will probably announce an extra office hour for next week.
- Suggestions of papers for project posted.
  - You can still choose a different paper, or do a project on your own research.
- Project Proposals Due Sunday Oct 29th 11:59pm.
- PSET1 Grades Out. Should have received mail with password & folder for feedback.

RECTIFIED BINOCULAR STEREO



Epipoles at infinity
Epipolar lines all parallel to the X axis

RECTIFIED BINOCULAR STEREO

Left

Right

L[x,y] matches to R[x',y]

Epipolar Lines are Horizontal

Why are we doing this ? Two equations tell us 3D position of point

RECTIFIED BINOCULAR STEREO

Left

Right

L[x,y] matches to R[x',y]

Epipolar Lines are Horizontal

Visibility Constraint: $x' \le x$

(object infront of camera)

RECTIFIED BINOCULAR STEREO

Left

Right

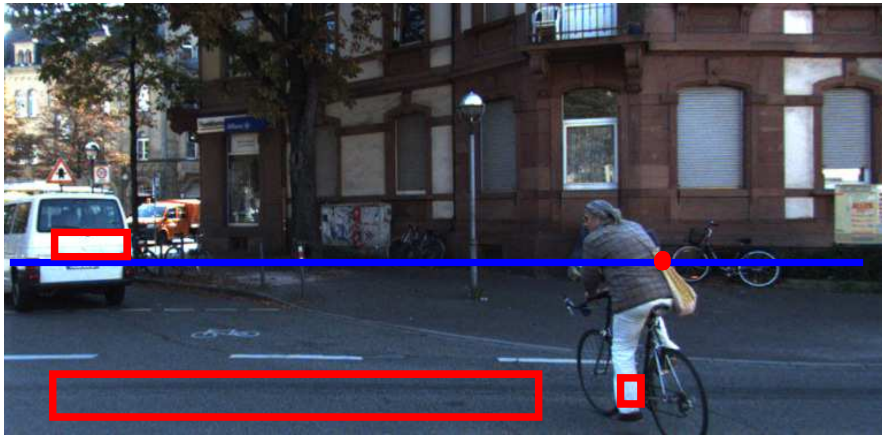
L[x,y] matches to R[x-d[x,y],y]

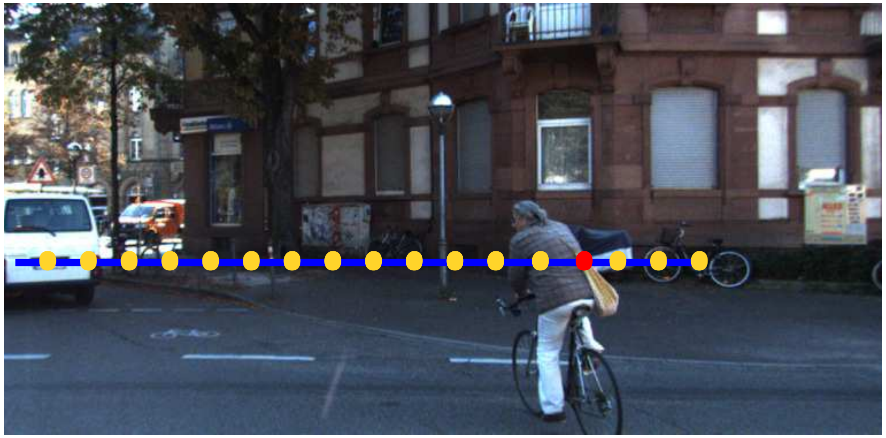
Epipolar Lines are Horizontal

d[x,y] >= 0 is called the "disparity map"

d[x,y] is inversely proportional to depth

RECTIFIED BINOCULAR STEREO

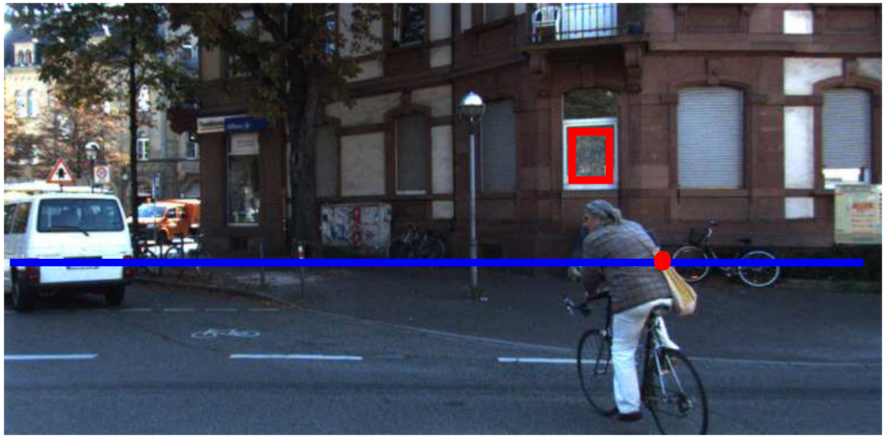


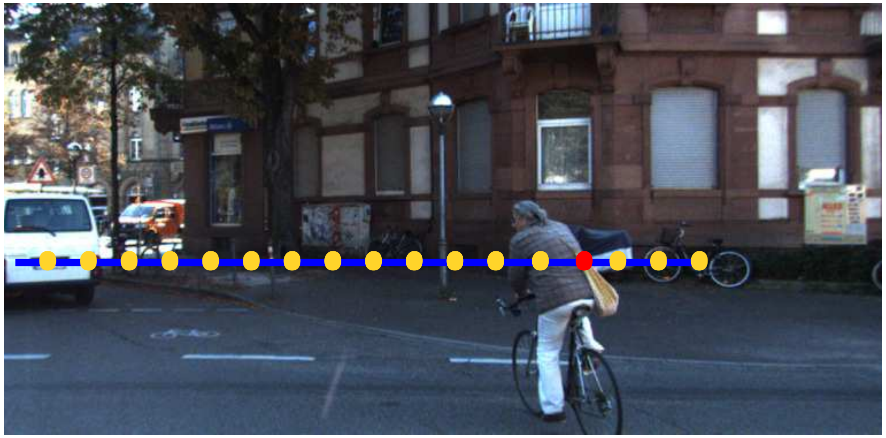


How do you find the correct match ?

-Smooth regions are ambiguous. Too many pixels on the right will look like a pixel on the left.

RECTIFIED BINOCULAR STEREO



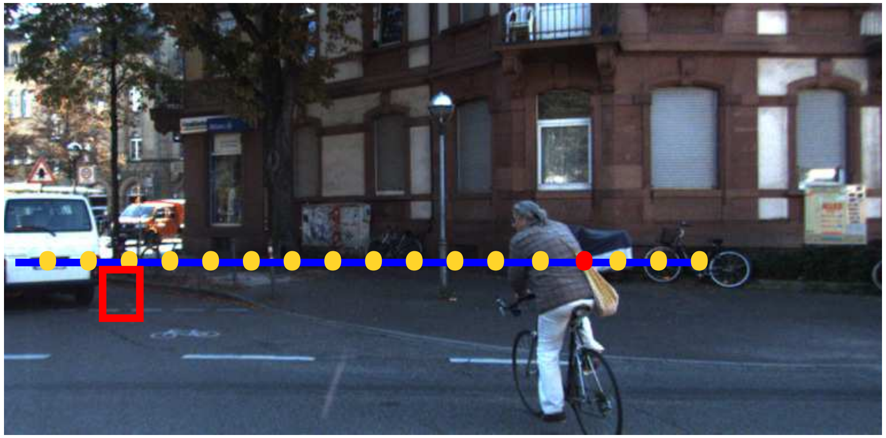


How do you find the correct match ?

-Smooth regions are ambiguous. Too many pixels on the right will look like a pixel on the left.

- Non lambertian regions are ambiguous. The correct pixel on the right will not look like the pixel on the left.

RECTIFIED BINOCULAR STEREO



How do you find the correct match ?

-Smooth regions are ambiguous. Too many pixels on the right will look like a pixel on the left.

- Non lambertian regions are ambiguous. The correct pixel on the right will not look like the pixel on the left.

- Occlusions: pixel on the left is NOT visible in the image on the right and vice-versa.

What is the right answer ?

d[x,y] = Value that the pixel [x,y] in the left image has moved by, even if it is not visible.
In other words, (x-d[x,y],y) should be the co-ordinate of the projection of that 3D surface point in the right image.

Useful because we want to eventually use it to estimate depth.

RECTIFIED BINOCULAR STEREO

How do you find the correct match ?

Consider a neighborhood

-Smooth regions are ambiguous. Too many pixels on the right will look like a pixel on the left.

- Non lambertian regions are ambiguous. The correct pixel on the right will not look like the pixel on the left.

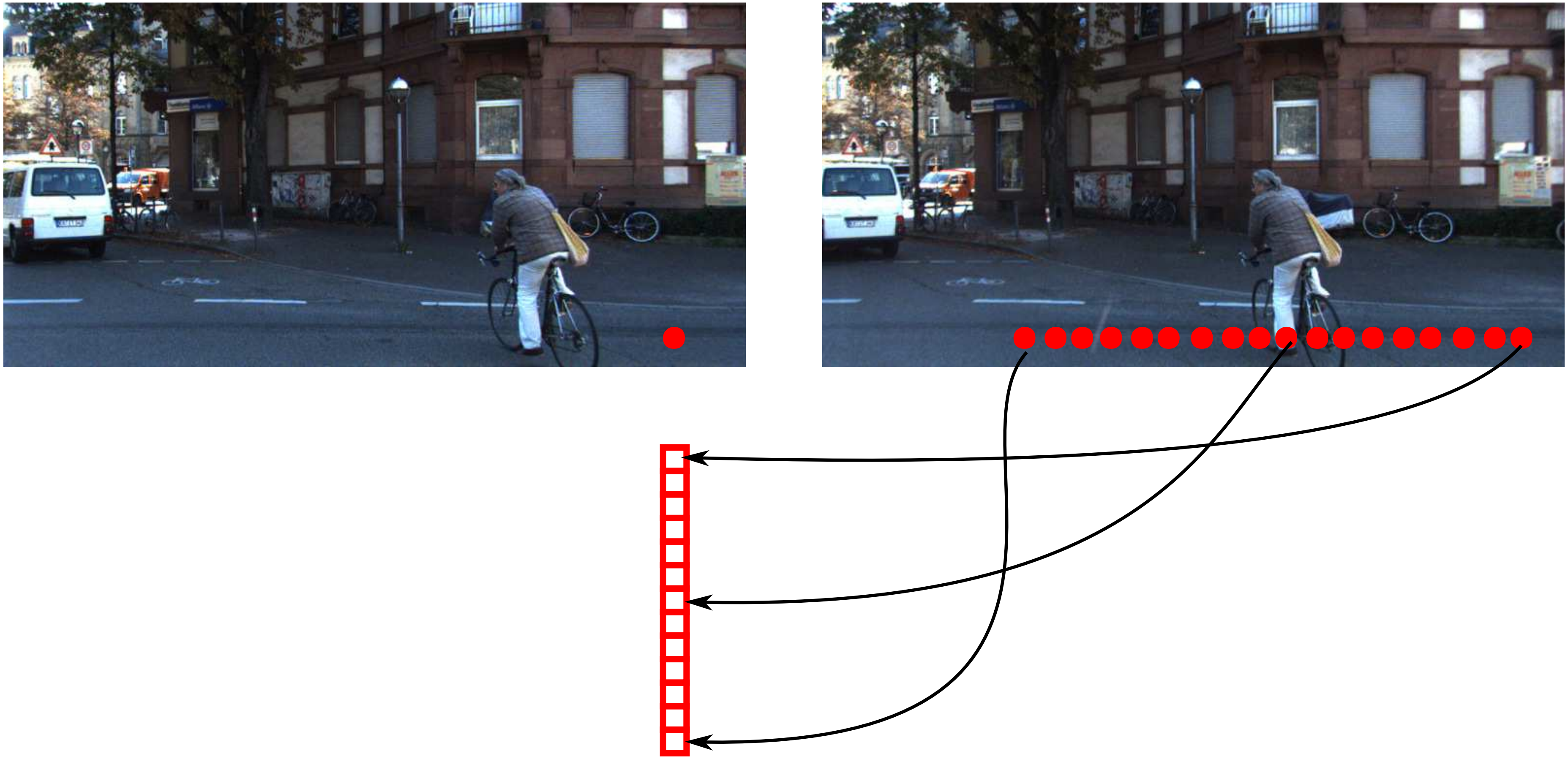
Robust features
(just look at sign of gradient
instead of magnitude)

- Occlusions: pixel on the left is NOT visible in the image on the right and vice-versa.

Last time: Matching with Census Transform

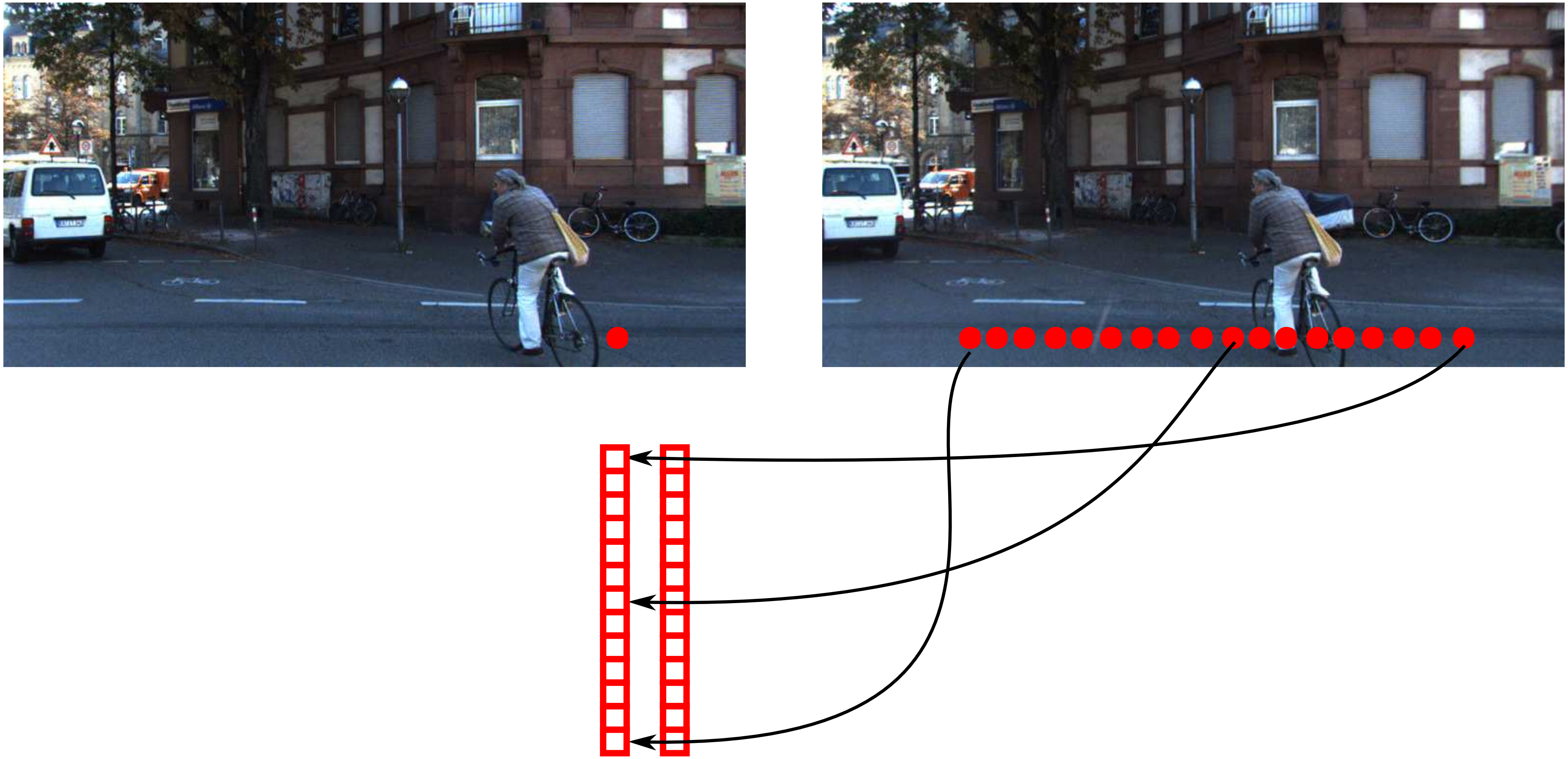
Just one of many 'robust features' / strategies for local matching.

COST VOLUME FILTERING



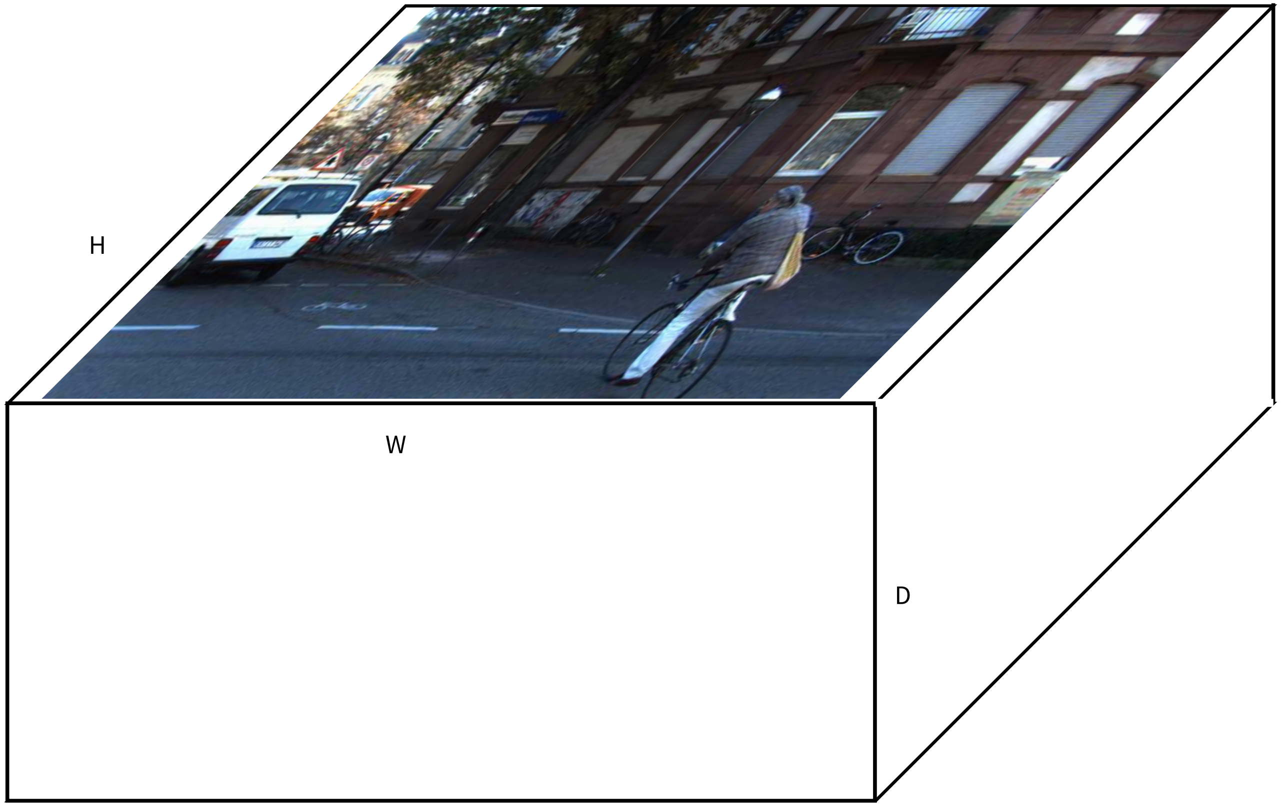
Let us say we believe max value of disparity is D-1
So by considering all possible matches, we are building a WxHxD "cost volume"

COST VOLUME FILTERING



Let us say we believe max value of disparity is D-1
So by considering all possible matches, we are building a WxHxD "cost volume"

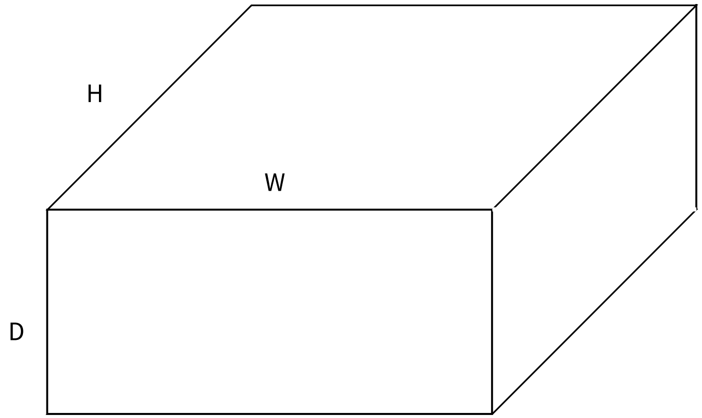
COST VOLUME FILTERING



C[x,y,d] measures the quality of the match between L[x,y] and R[x-d,y]

Let us say we believe max value of disparity is D-1
So by considering all possible matches, we are building a WxHxD "cost volume"

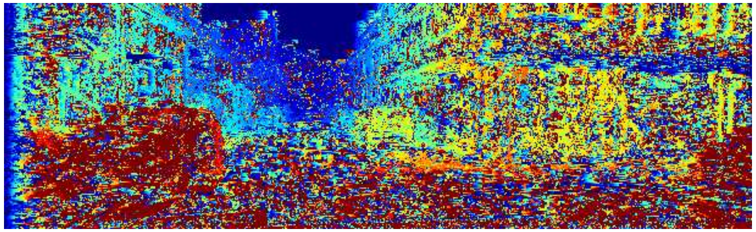
COST VOLUME FILTERING



C[x,y,d] measures the quality of the match between L[x,y] and R[x-d,y]

In problem set 3, you simply compute the best match independently at each [x,y]

$$d[x,y] = \arg\min_{d} C[x,y,d]$$

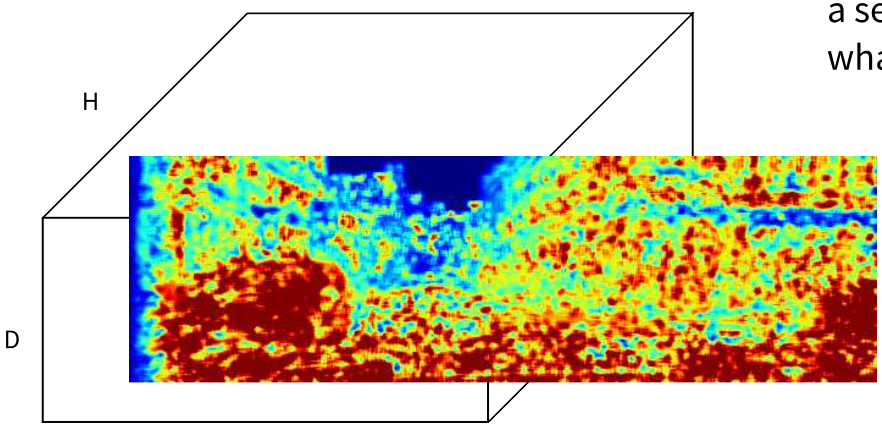


But that still gives us pretty noisy disparity maps

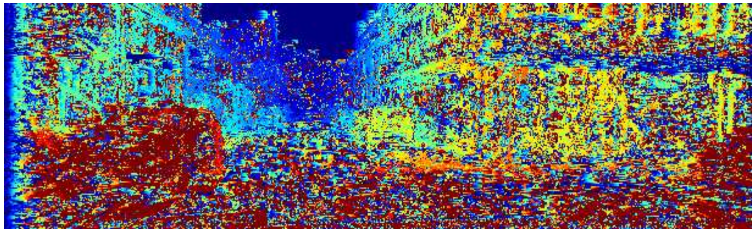
We've seen noise before. Smoothing helps.
We could just smooth the disparity map ?

COST VOLUME FILTERING

The errors in the disparity map can often be high magnitude. In a smooth region or with repeated texture, there may be a second seemingly good match very far away, and that's what arg min chooses.



$$d[x, y] = \arg\min_{d} C[x, y, d]$$



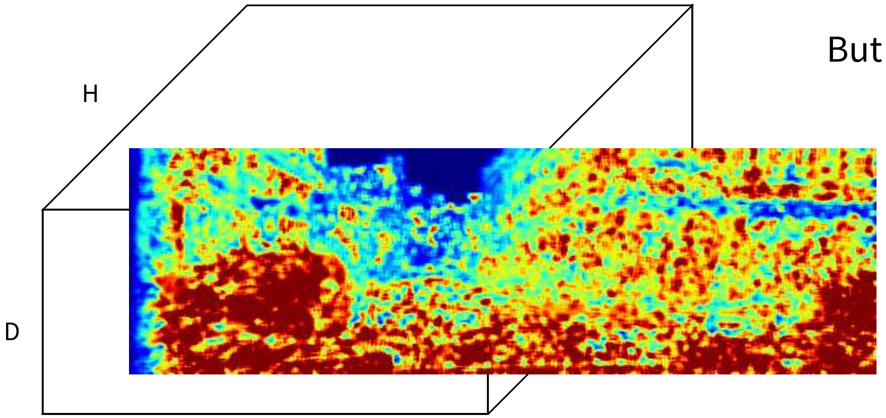
But that still gives us pretty noisy disparity maps

We've seen noise before. Smoothing helps.
We could just smooth the disparity map ?

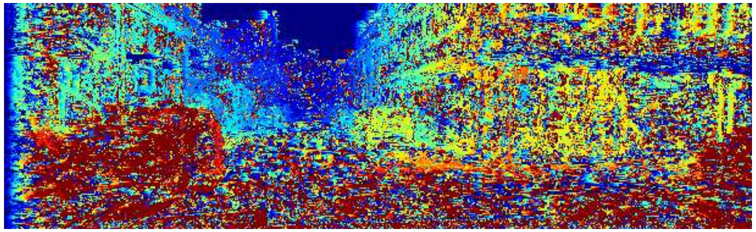
COST VOLUME FILTERING

We want to express the fact that we expect our disparity map to be smooth.

But do it before we compute the arg min below.



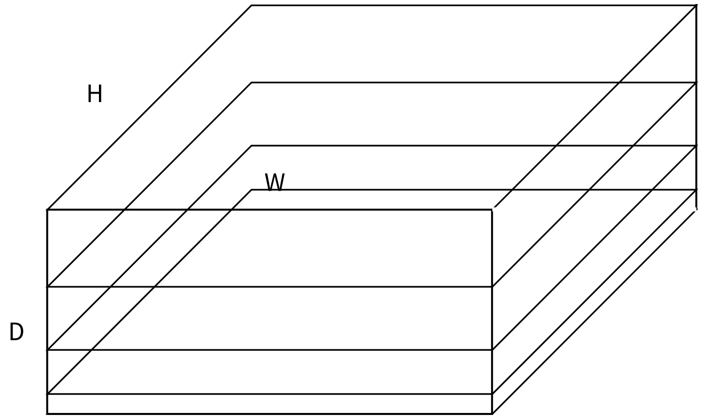
$$d[x,y] = \arg\min_{d} C[x,y,d]$$



But that still gives us pretty noisy disparity maps

We've seen noise before. Smoothing helps.
We could just smooth the disparity map ?

COST VOLUME FILTERING

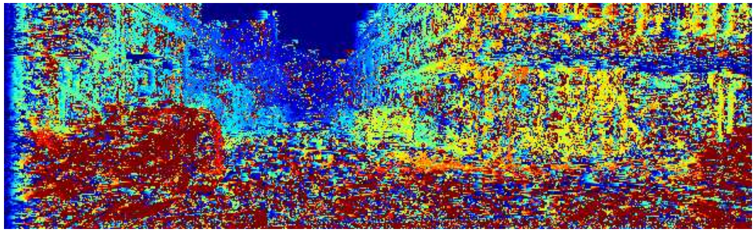


Possible Solution: Smooth the cost volume !

Take each slice of the cost volume, and smooth it.
Expresses the fact that if [x,y] and [x-d,y] match, then so should [x+1,y] with [x+1-d,y]; [x,y+1] with [x-d,y+1]; [x-1,y] with [x-1-d,y]

Take arg min AFTER smoothing

$$d[x,y] = \arg\min_d \overline{C}[x,y,d]$$



But that still gives us pretty noisy disparity maps

We've seen noise before. Smoothing helps.
We could just smooth the disparity map ?

COST VOLUME FILTERING

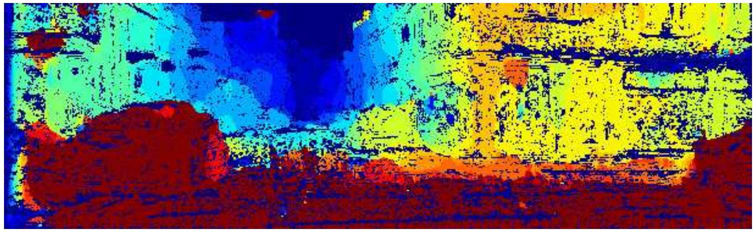
Possible Solution: Smooth the cost volume !

Take each slice of the cost volume, and smooth it.
Expresses the fact that if [x,y] and [x-d,y] match, then so should [x+1,y] with [x+1-d,y]; [x,y+1] with [x-d,y+1]; [x-1,y] with [x-1-d,y]

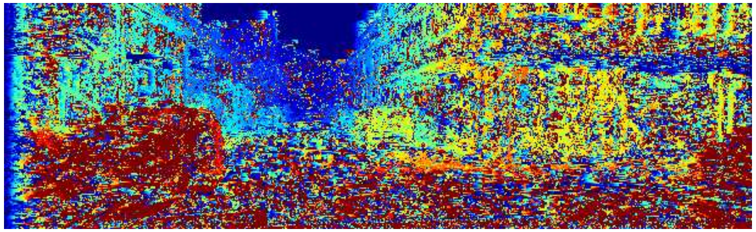
Take arg min AFTER smoothing

$$d[x,y] = \arg\min_d \overline{C}[x,y,d]$$

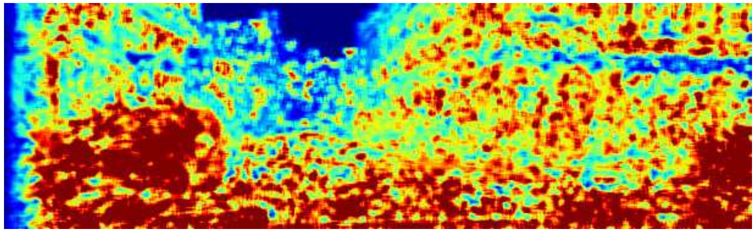
Smoothing Cost Volume



Original Disparity Map

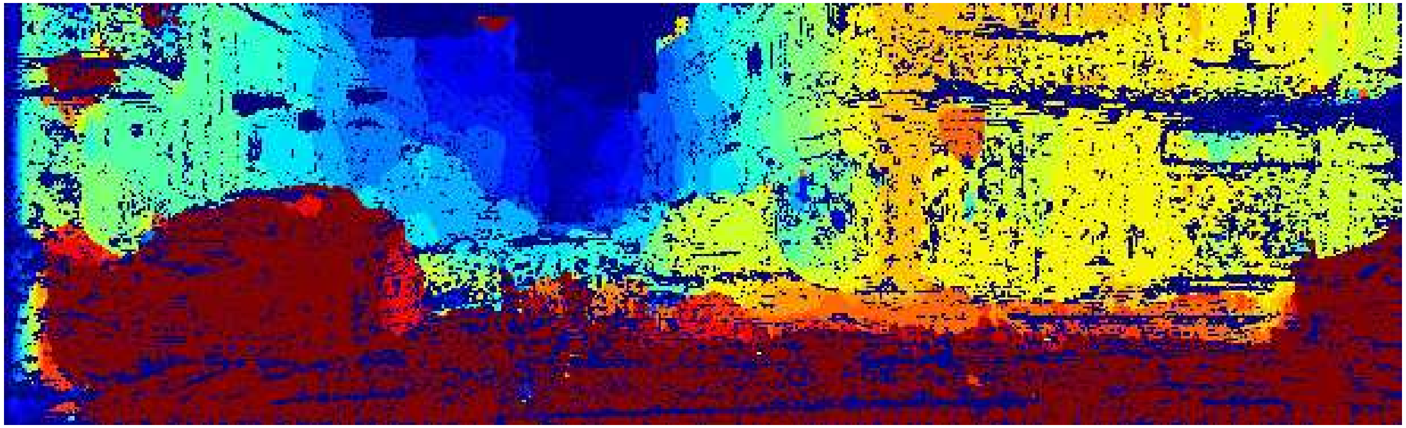


Smoothing Disparity



COST VOLUME FILTERING

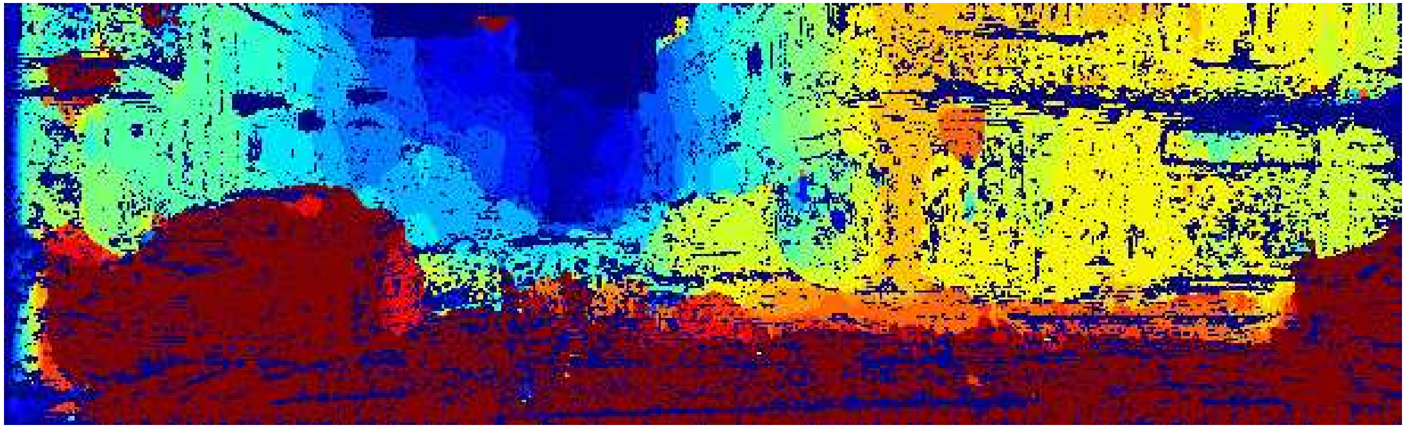
More Smoothing
Less Noise
Blurrier Edges



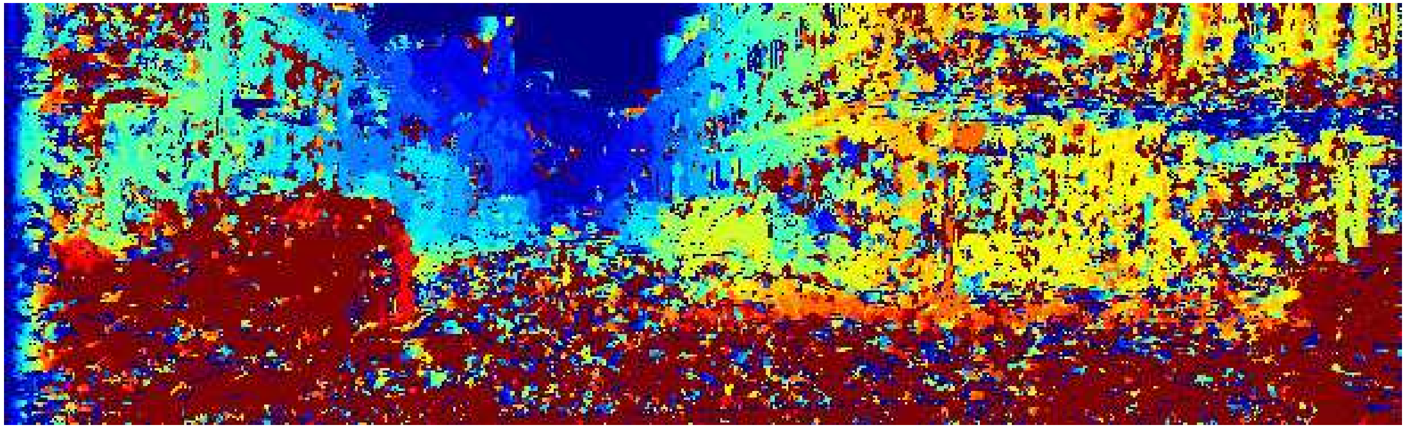
Here, "blurrier" means the location of disparity discontinuities, i.e. the contours, get spread out. It does not cause a more gradual change in the disparities themselves.

COST VOLUME FILTERING

More Smoothing
Less Noise
Blurrier Edges

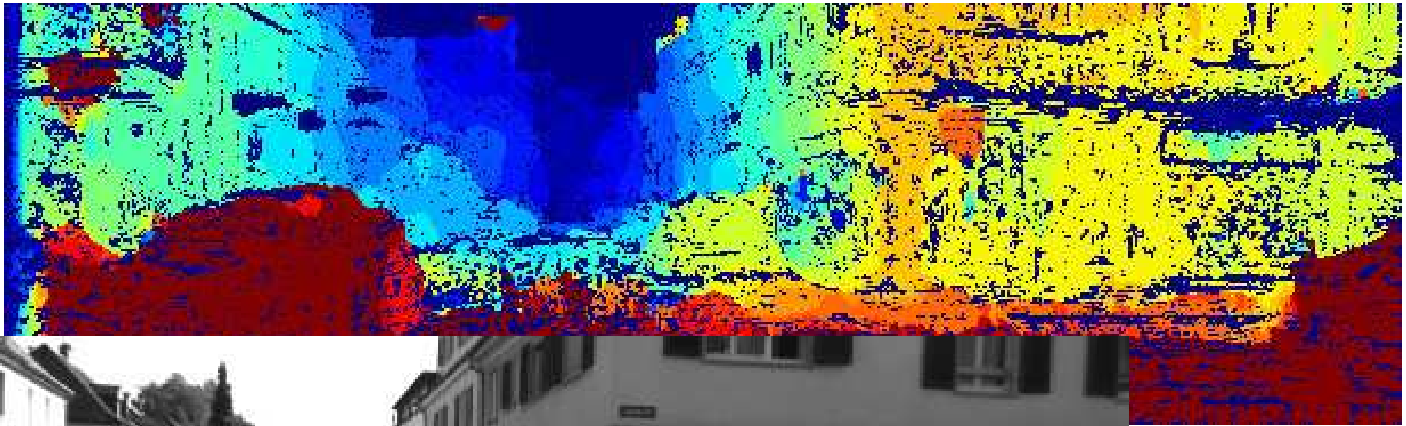


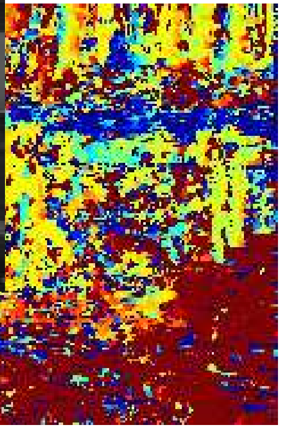
Less Smoothing
More Noise
Sharper Edges



COST VOLUME FILTERING

More Smoothing
Less Noise
Blurrier Edges

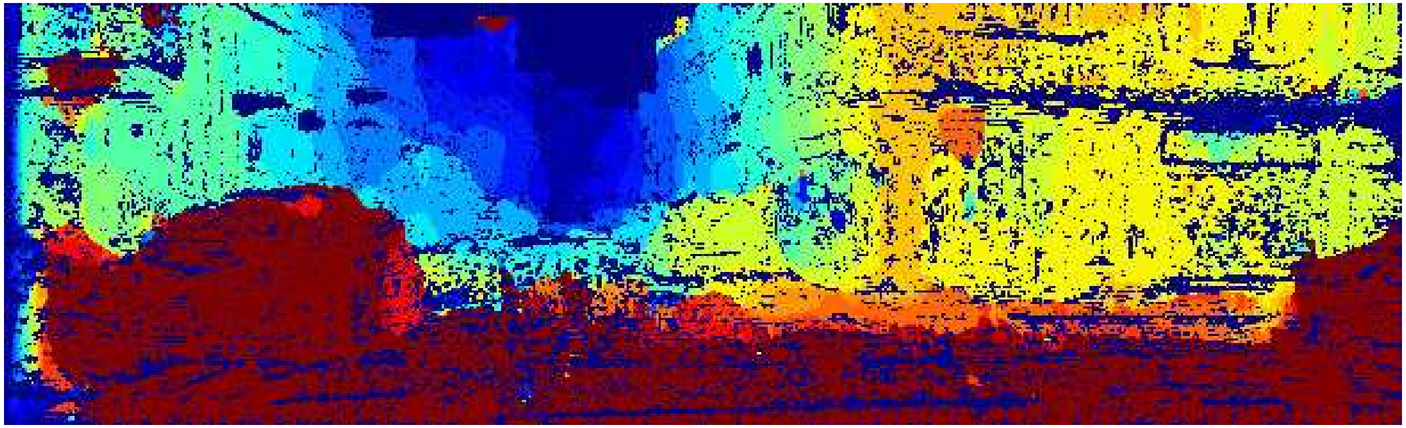




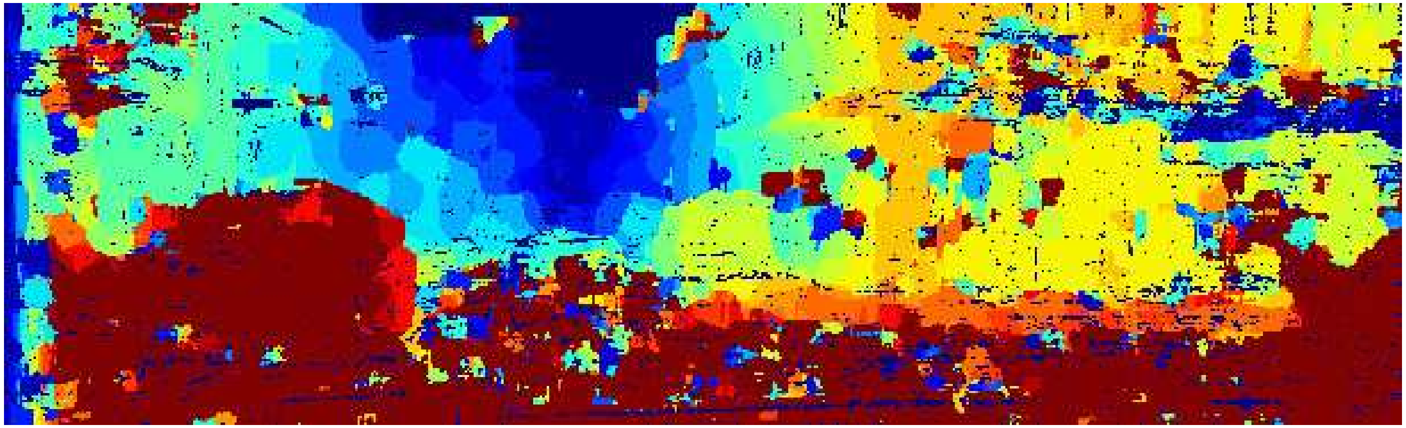
Less
More
Sharper Edges

Bilateral Filtering, guided by left image

COST VOLUME FILTERING



Other variants include, thresholding the cost before smoothing.



GLOBAL OPTIMIZATION

- Going back, we want to express the fact that our disparity map is smooth.
- Cost volume filtering is an ad-hoc way of doing that.
  - Still making independent decisions at each pixel.
- Averaging each disparity level promotes disparity maps where values are "equal" not close.
  - If $C[x, y, d]$ is a good match, then $C[x+1, y, d \pm 1]$ gets no benefit from filtering.
  - Not good for slanted surfaces.
- Could be fixed by smoothing

$$\min_{\delta=\{-1,0,1\}} C[x, y, d+\delta]$$

- But generally, would prefer expressing this as optimizing a well-defined cost.

GLOBAL OPTIMIZATION

$$d = \arg\min_d \sum_n C[n, d[n]] + \lambda \sum_{(n,n') \in \mathbf{E}} S(d[n], d[n'])$$

- $n = [x, y]^T$ for pixel location.
- C is cost-volume as before. Gives us "local evidence"
- $\mathbf{E}$ is a set of all pairs of pixels that are "neighbors" / adjacent in some way.
  - Can include all un-ordered pairs of pixels with $[(x, y), (x-1, y)]$ and $[(x, y), (x, y-1)]$ (four connected)
  - Or diagonal neighbors as well.
- S is a function that indicates a preference for $d[n]$ and $d[n']$ to be the same.

GLOBAL OPTIMIZATION

$$d = \arg\min_d \sum_n C[n, d[n]] + \lambda \sum_{(n,n')\in\mathbf{E}} S(d[n], d[n'])$$

- S is a function that indicates a preference for $d[n]$ and $d[n']$ to be the same.

- Choice 1:
  - 0 if $d[n'] = d[n]$, 1 otherwise.
- Choice 2: $|d[n'] - d[n]|$
- Choice 3:
  - 0 if $d[n'] = d[n]$
  - T_1 if $|d[n'] - d[n]| < \epsilon$
  - T_2 otherwise.

Note that this is a discrete minimization. Each $d[n] \in \{0, 1, \ldots D - 1\}$.

How do we solve this ?

GLOBAL OPTIMIZATION

$$d = \arg\min_d \sum_n C[n, d[n]] + \lambda \sum_{(n,n')\in\mathbf{E}} S(d[n], d[n'])$$

One approach: Iterated Conditional Modes

- Begin with $d_0 = \arg min_d C[n, d[n]]$
- At each iteration t, compute d_{t+1} from d_t, by solving
  for each pixel in d_{t+1} assuming neighbors have values from d_t.

$$d_{t+1}[n] = \arg\min_{d_n} C[n, d_n] + \lambda \sum_{(n,n')\in\mathbf{E_n}} S(d_n, d_t[n'])$$

So for each pixel,

- Take matching cost.
- Add smoothness cost from its neighbors, assuming values from previous iteration.
- Minimize.

Does it converge ?

To a global optimum ?